Universitat de Lleida

TREBALL FINAL DE GRAU

ESCOLA POLITÈCNICA SUPERIOR
UNIVERSITAT DE LLEIDA
INSPIRING THE FUTURE

Estudiant: Chaimae El Morabet Lachkar

Titulació: Grau en Enginyeria Informàtica

Títol de Treball Final de Grau: **ColdBox: Gestió de l'estoc en una cooperativa de fruita**

Director/a: **Fernando Cores**

Presentació

Mes: Setembre

Any: 2019

Contenido

Tabla de ilustraciones

1 Introducción

El trabajo gira entorno a la idea de una cooperativa agrícola. En estas acostumbran a tener un control de stock, ya sea más o menos informatizado.

En este caso, se usa como referencia el sistema de gestión que tienen en una cooperativa situada en Aitona. En ella tienen un sistema informático en el cual se introducen los kilos que entran y los que salen durante la jornada laboral.

A parte del sistema informático en ella siempre hay un operario que al final de la jornada laboral hace el recuento manual de kilos de fruta restantes para el siguiente día. El resultado de este recuento se tiene que pasar a otros operarios la mañana del día siguiente para así poder planificar el uso de los quilos.

Acostumbran a hacer una aproximación diaria ya que si no sería muy complicado programar toda una jornada laboral sabiendo los quilos con los que se trabajarán a pocas horas de que este empiece.

De aquí nace la idea principal del trabajo, que recae en el diseño y desarrollo de una aplicación móvil en la cual poder introducir los datos y que estén a disposición de todos los usuarios de la aplicación que quieran consultarlos.

Detallar un poco más la funcionalidad de la aplicación y explicar el montaje de la báscula.

1.1 Objetivos

- Diseñar y desarrollar una aplicación móvil, tanto para Android como iOS, para gestionar de forma sencilla datos generados por la cooperativa.
- Simular el proceso de gestión manual de la cooperativa desde una perspectiva más tecnológica.
- Usar un lenguaje de programación multiplataforma que permita abastecer un mayor número de usuarios.
- Diseñar y ensamblar una báscula inteligente para simular el proceso de entrada/salida de fruta.

1.2 Planificación del proyecto

Como en todo buen proyecto tiene que haber una buena planificación temporal para realizarlo. Este proyecto tiene una durada de 6 meses, por lo tanto, se tiene que hacer una planificación de tareas a realizar y así tener un control sobre el tiempo que podemos invertir en cada tarea en un orden establecido.

1.2.1 Definición de Tareas

- Diseño

El diseño de la aplicación va a ser pensado antes de la implementación de la aplicación. Cabe decir que no se sabe aún como va a ser la aplicación ni que componentes la van a componer. La aplicación se va a diseñar con ayuda de bocetos hechos a mano o Wireframes [18].

Se tendrá que tener en cuenta que para el diseño de la aplicación, ya se tiene alguna idea de cómo podría ser, si estas ideas serán posibles de desarrollar ya que puede que por conocimiento puede que no sepa implementarlas. Así pues, el diseño podrá ir variando a medida que se vaya implementando la aplicación.

- Diferentes partes de implementación

Este proyecto tendrá dos partes claramente delimitadas a la hora de hablar de la implementación.

Por una parte, tendremos la implementación de la aplicación y por otra la de la báscula inteligente.

- Validación y pruebas

Las validaciones y las pruebas se harán a medida que se vaya implementado la aplicación. Algunas pruebas las haré yo, pero se ha pensado en pedir ayuda a otros usuarios potenciales para ver si realmente es funcional y usable por personas que tendrán su primer contacto con ella.

Estas pruebas y validaciones también me servirán para recoger FeedBack de la aplicación y así poder hacer las modificaciones oportunas ya sean en diseño o en funcionalidad.

- Escritura memoria

Durante la realización del proyecto, se tendrá que ir documentado todo el proceso tecnológico. Esta documentación da como resultado la memoria del proyecto.

1.2.2 Diagrama de Gantt

Para realizar esta planificación he usado *GanttProject***¡Error! No se encuentra el origen de la referencia.**. Es un software gratuito que permite la elaboración de diagramas de Gantt de una forma simple y sencilla.

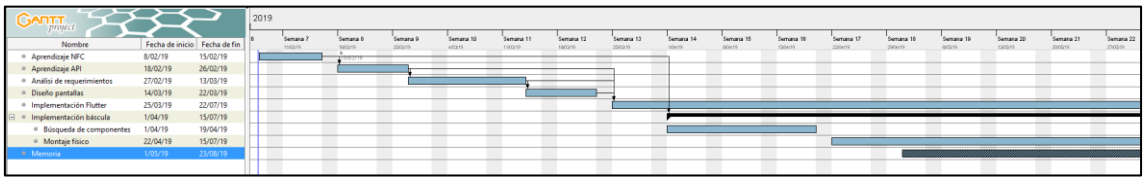

Ilustración 1 Diagrama de Gantt (1/2)

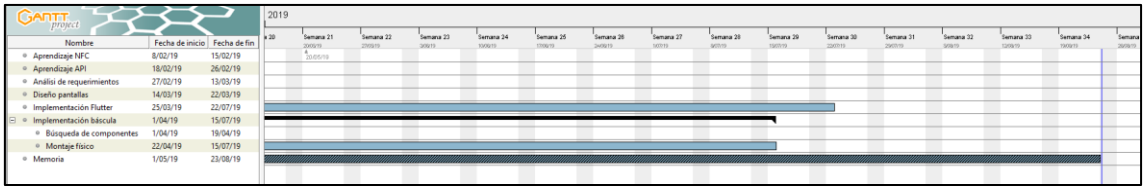

Ilustración 2 Diagrama de Gantt (2/2)

Tal y como se puede observar en las dos ilustraciones anteriores, Ilustración 1 e Ilustración 2, al principio del proyecto hay planificadas varias tareas de recerca.

Después de cursar dos asignaturas donde se estudia la plataforma Android no he tenido la oportunidad de aprender cómo funciona una antena NFC que llevan los dispositivos móviles. También he incluido todo el proceso de montaje de la báscula inteligente ya que es una tarea importante del proyecto.

La memoria se tendría que haber puesto a lo largo de toda la planificación temporal, para así poder ir documentando todo el transcurso del proyecto por partes, pero como se puede observar, no ha sido el caso y ha sido la última tarea a realizar y consecuentemente, la tarea critica.

1.3 Presupuesto del proyecto

1.3.1 Software

En la parte de software no ha habido ningún tipo de coste ya que se han usado programas de libre distribución. Por lo tanto, el coste será contado por las horas que se han dedicado a los diseños y la implementación de la aplicación.

Concepto	Horas	Precio
Diseño aplicación	7	350 €
Implementación aplicación	200	10.000€
Implementación báscula inteligente	20,5	1.025 €

Tabla 1 Coste proyecto ColdBox

El presupuesto se ha valorado como si fuera un trabajo profesional y se han contado las horas trabajadas a 50€/hora. Es un precio bastante alto, pero es el que hoy en día se cobra de media en cualquier empresa. El coste total es de 11.375€. Es un precio valorado por la cantidad de funcionalidades implementadas.

1.3.2 Hardware

En la siguiente tabla podemos observar el coste de cada componente usado para la realización del montaje físico del proyecto.

Concepto	Unidades	Precio
Raspberry Pi - Caja Raspberry Pi B+ y Pi2, color negro	1	8,86 €
NorthPada Europa Cargador 5V 2A 2000mA Micro USD para Raspberry Pi Model A y B & Banana Pi - Pi Model B+	1	8,99 €
Raspberry Pi 2 Model B - Placa base (ARM Quad-Core 900 MHz, 1GB RAM, 4 x USB, HDMI, RJ-45)	1	40,20 €
Kingston SDC10/16GB - Tarjeta microSD de 16 GB, negro	1	4,29 €
Kit 40 Jumpers de 20cm Macho/Macho	1	0,49 €
Módulo WiFi NodeMCU Lua Lolin V3 ESP8266	1	7,49 €
Sensor electrónico de peso portátil de Haljia, sensor de peso de celda de carga 20Kg	1	9,99 €
HX711 Módulo de sensor de pesaje de convertidor analógico a digital (ADC)	1	5,53 €
AptoFun RC522 - Lector de Tarjetas RFID para Arduino y Raspberry Pi	1	7,50 €
Breadboard Placa prototipos sin Soldadura para Rasperry Pi Arduino	1	6,00 €
Total	**10**	**99,34 €**

Tabla 2 Coste montaje físico

En la parte física el coste total ha sido de 99,34€, como se puede ver en la Tabla 2. No es un precio excesivo ya que solamente es un prototipo y se han usado materiales baratos per a la realización del prototipo. En condiciones normales el precio sería mucho más elevado ya que la maquinaria sería bastante diferente.

1.4 Estructura del documento

El resto de la memoria del proyecto está organizado en los siguientes capítulos.

En el capítulo *2Análisis de viabilidad* teorizo sobre la viabilidad de mi proyecto en la vida real, realizando un análisis DAFO i un estudio de mercado.

En el capítulo *Análisis y diseño* explico la tecnología usada para realizar la parte práctica del trabajo: el entorno de desarrollo y las herramientas de desarrollo que uso. También detallo las funcionalidades y requerimientos que tendrá que tener tanto el software como el hardware.

En el capítulo *Decisiones e implementación* explico las decisiones tomadas a la hora de implementar la aplicación y su implementación juntamente con la implementación de la báscula inteligente.

En el capítulo Conclusiones expongo mis conclusiones respecto el proyecto realizado y posibles trabajos futuros.

2 Análisis de viabilidad

2.1 Estudio de viabilidad

Observando algunas aplicaciones, las cuales tratan el tema de stock, he podido observar que básicamente eran todas muy similares.

Estas son algunas de las aplicaciones que se han encontrado buscando en la *Play Store de Google.*

2.1.1 Stock Controller – inventories

A simple vista es una aplicación de control de inventario de productos a nivel de tienda o almacén. Se podría comparar bastante a lo que yo quiero en mi aplicación, ya que tiene un inventario de productos, también tiene el almacén en el que se encuentra situado, que serían las cámaras de refrigeración de una cooperativa, y tiene un panel de estadísticas de inventario.

Esta aplicación contiene anuncios en ella, y se tendría que pagar por la versión Premium para poder quitarlos, cosa que encuentro molesta en el uso de una aplicación.

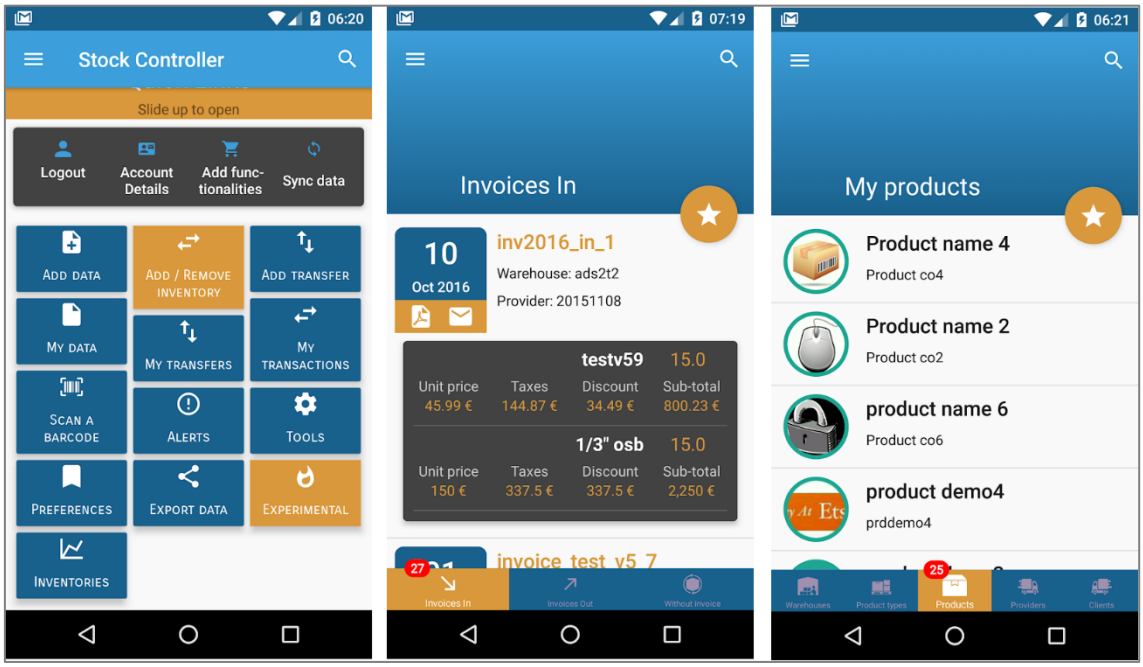

Ilustración 3 App 1

2.1.2 Stock and Inventory Simple

Muy similar al anterior en funcionalidad, solo que este incluye gestión documental de los movimientos de stock, ya sean movimientos de entrada o salida. Estos documentos se pueden exportar e importar en formato Excel. Esto es un punto bastante interesante ya sería una buena mejora del proyecto en el futuro porque es de mi interés el hecho de poder exportar e importar movimientos con documentos Excel.

También cabe decir, que desde mi punto de vista, es una aplicación poco atractiva visualmente, creo que el diseño es un punto importante en la usabilidad de las aplicaciones.

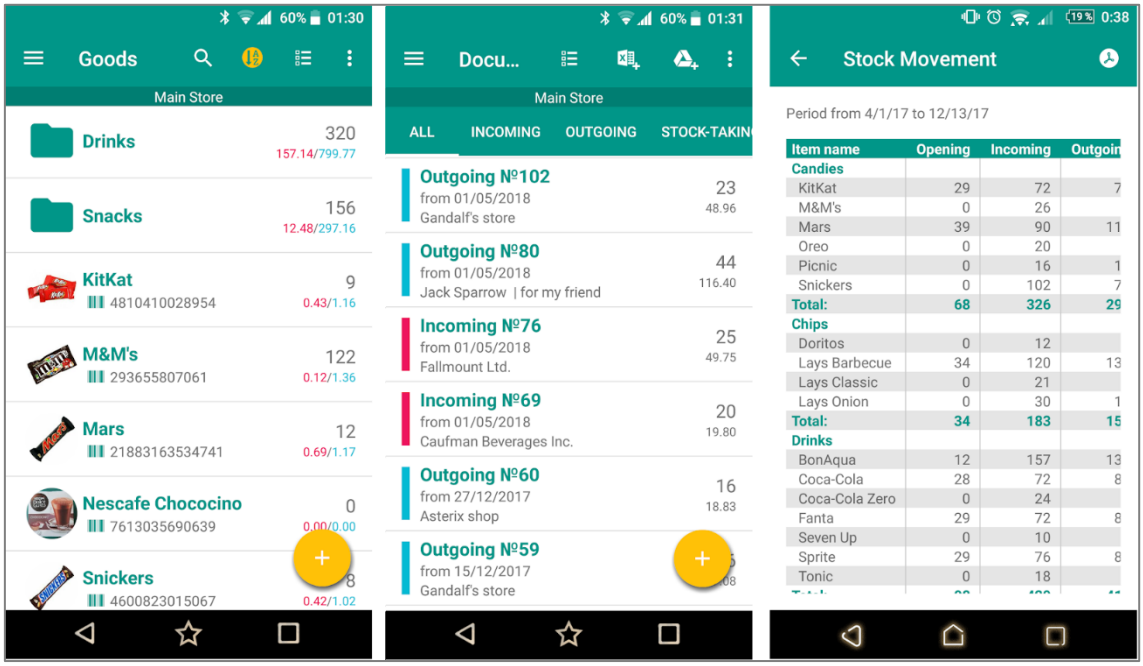

Ilustración 4 APP 2

2.1.3 TapPOS Inventry Sales manager

Esta aplicación tienes las mismas funcionalidades que las anteriores. A parte de las funcionalidades que tienen las dos anteriores, esta tiene el añadido de poder imprimir o enviar el recibo, cosa que las otras dos aplicaciones no tienen.

También cabe remarcar que creo que el diseño de esta aplicación es mejor que el de las anteriores ya que es mucho más visual al contener dibujos.

Pero creo que el hecho de tener que pagar por ella es negativo. Es normal que las aplicaciones valgan dinero, pero creo que tener que pagar 3,99€ por una única funcionalidad diferente, ya que las otras que tiene esta aplicación las tienen las anteriores aplicaciones, creo que es caro.

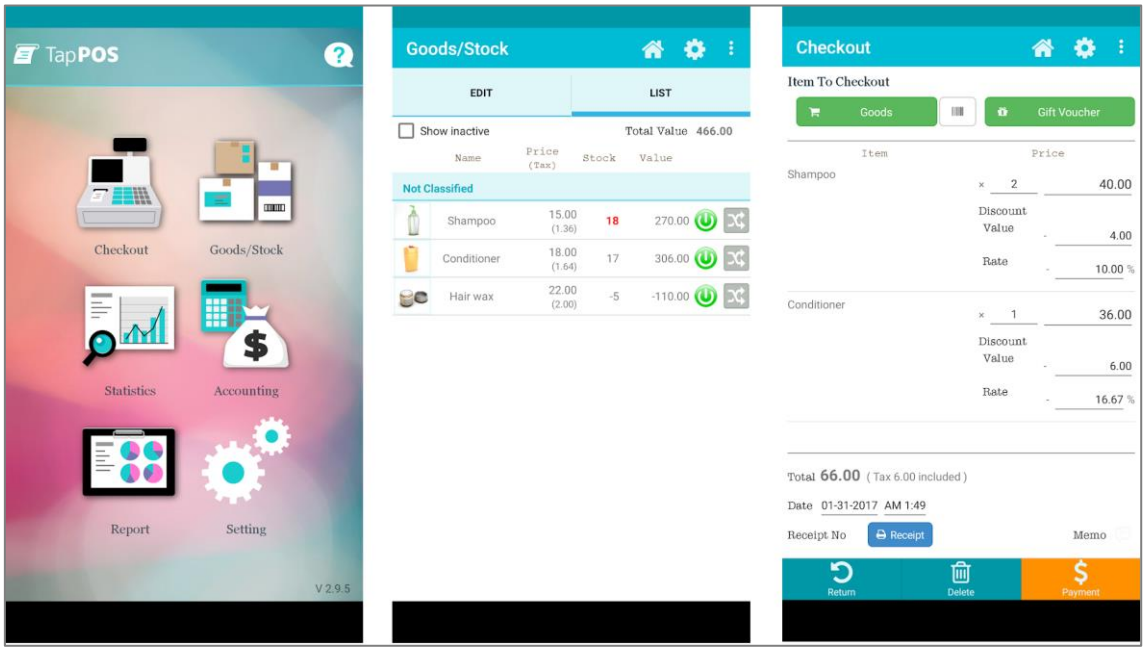

Ilustración 5 App 3

2.1.4 My Stock Inventory Mobile Cloud barcode scanner

Esta última es la que parece más completa comparada con las anteriores, ya que es una aplicación que uno puede personalizarla siempre y cuando page los costes añadidos que se le presupuestarían.

Sus funcionalidades son muy parecidas, para no decir las mismas, pero como bien he dicho, mediante pagos se le puede añadir más. La principal funcionalidad que se le puede añadir, ya que es una general, es la de migrar documentos Excel y compartirlos mediante Google sheets. Pero como bien he dicho anteriormente, se puede personalizar toda la aplicación, aunque el pago por esto sería distinto que el de añadir la migración de documentos.

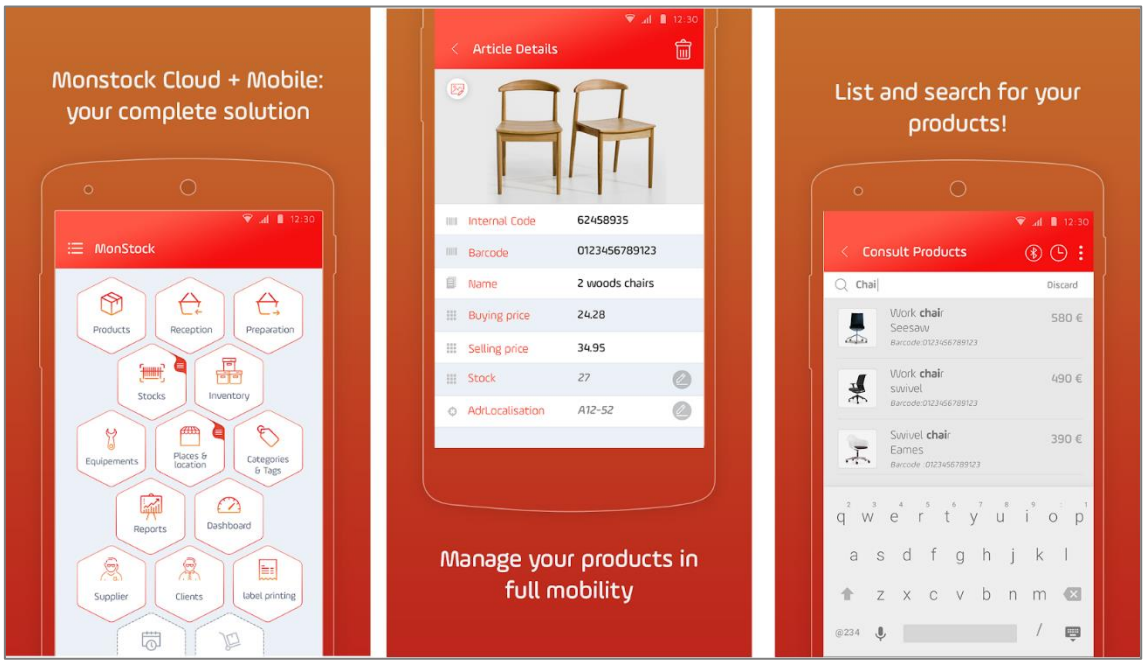

Ilustración 6 App 4

2.2 Análisis DAFO

El Análisis DAFO es un método de planificación estratégica para evaluar las Debilidades, Amenazas, Fortalezas y Oportunidades de un proyecto. Consiste en un análisis que diferencia entre los factores internos (fortalezas y debilidades) de una organización y los factores externos de esta (Oportunidades y amenazas). Se trata de especificar el objetivo de un proyecto y la identificación los factores internos y externos que son favorables y desfavorables para alcanzar este objetivo.

2.2.1 Debilidades

- Tal y como podemos ver día a día, las aplicaciones suelen tener diseños en un principio gráficos muy elaborados. Esto es debido a que las aplicaciones, con sólo echar un vistazo, entran por los ojos al usuario. Esto quiere decir que, aunque una aplicación sea mejor que otra, la que tiene el mejor diseño gráfico suele ser la que tiene más éxito aunque no sea tan buena. Por lo tanto, puedo considerar una debilidad el no tener a mi disposición un diseñador gráfico.
- Nunca he trabajado con la tecnología NFC y no sé hasta qué punto podría desarrollar las funcionalidades que le corresponden.
- Tengo muchas carencias en el ámbito de la electrónica y realizar el prototipo final de la báscula inteligente puede ser bastante costoso.

2.2.2 Amenazas

- La principal amenaza es el rechazo por parte de los usuarios finales de la aplicación ya que podrían no verle utilidad esta aplicación.
- Es muy probable que no tenga gran recibimiento en el mercado ya que muchas empresas pueden tener sus aplicaciones privadas. Las que han desarrollado con un equipo y con previo estudio de la empresa y sus necesidades.

2.2.3 Fortalezas

- He tenido la oportunidad de poder experimentar bastante con el entorno "*Flutter"* aunque todavía me faltan muchas cosas por aprender hacer esto es una fortaleza ya que no es necesario que dependa de nadie para implementar la aplicación, aunque hay temas específicos que necesitaré documentarme.
- Me gusta mucho diseñar y crear aplicaciones.
- Todas las herramientas que necesito para desarrollar la aplicación son totalmente gratuitas y por tanto no me supondría ningún coste económico.

2.2.4 Oportunidades

- La aplicación móvil se desarrollará siendo multiplataforma, esto significa que será una aplicación valida tanto para Android como iOS. Para que la aplicación será compatible con dispositivos iOS, esta se tendrá que compilar en un ordenador con sistema operativo macOS. De esta manera, se podrá llegar a más usuarios.
- Dentro del mundo hortofrutícola esta aplicación podría ser de gran interés ya que está enmarcada en este mundo y, además, está diseñada pensando en una cooperativa de fruta.
- Solo sería necesario que una empresa del sector estuviera interesada en probar la aplicación, y así hacer ver al resto de empresas, en verso el resultado obtenido por esta, lo útil que puede llegar a ser y así conseguir que otras empresas también quisieran hacer uso de la aplicación.

2.3 Tecnologías utilizadas

En este capítulo se explican las tecnologías, tanto software como hardware, utilizadas en el proyecto. Estas se usan en la construcción de la báscula inteligente y en la implementación de la aplicación.

Como que la báscula inteligente es un componente físico del proyecto, se han tenido que usar elementos hardware para su construcción y correcto funcionamiento.

2.3.1 Tecnologías Hardware

2.3.1.1 Placa ESP8266

Esta placa es la que se encarga de todas las operaciones relacionadas con la báscula inteligente y la conexión de esta con la base de datos.

NodeMCU es una placa de desarrollo totalmente abierta, a nivel de programario y de maquinaria. Igual que pasa con Arduino, a NodeMCU todo está dispuesto para facilitar la programación de un microcontrolador o MCU (del inglés MicroController Unit).

Para el prototipo de bascula inteligente, se ha usado una placa electrónica llamada ESP8266 versión 1.0 basada en NodeMCU. Esta es muy similar a un Arduino, encara que una de las principales diferencias, y principal motivo de porque se ha elegido esta placa es que incorpora un chip de conexión Wifi.

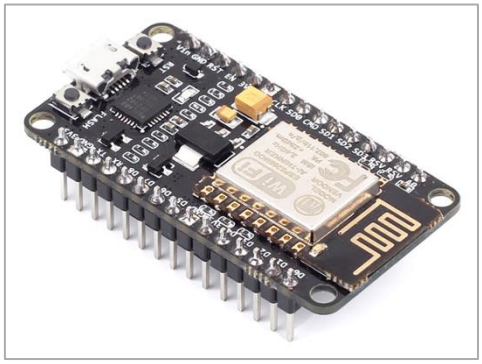

Ilustración 7 Placa ESP8266 NodeMCU

2.3.1.2 Celda de carga

Una celda de carga, sirve para calcular el peso de un objeto y es un transductor capaz de convertir una fuerza en una señal eléctrica, esto lo hace a través de uno o más galgas internas que posee, configuradas en un puente Wheatstone.

Este puente es un circuito eléctrico que se usa para medir resistencias desconocidas mediante el equilibrio de los brazos del puente.

Hay diversos tipos de celdas de carga, en distintos modelos, el que se usara para este proyecto es el que se muestra en la Ilustración 8. Usaremos una celda de carga de 20kg, aunque también existen otros modelos que pueden soportar más o menos quilos.

Ilustración 8 Celda de carga

2.3.1.3 Módulo HX711

El módulo HX711 (Ilustración 9) es la interfaz de comunicación entre la celda de carga, explicada anteriormente, y el microcontrolador permitiendo de esta manera una fácil lectura del peso.

Internamente se encarga de la lectura del puente Wheatstone de la celda de carga convirtiendo así la lectura analógica a digital a través de su conversor A/D interno de 24 bits.

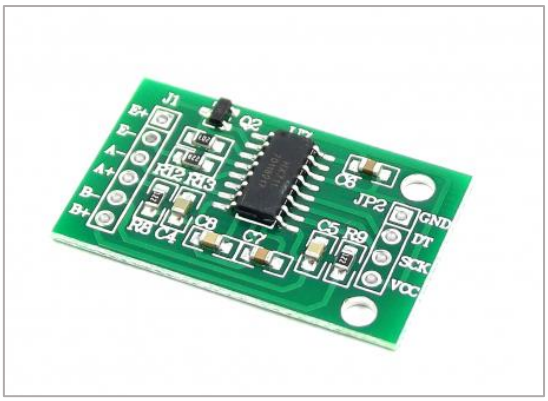

Ilustración 9 Módulo HX711

2.3.1.4 Lector RFID RC522

El lector RFID (Radio frequency identification) tiene un funcionamiento que consiste en pasar un TAG, cerca del lector. Este TAG tiene la capacidad de enviar información al lector. Esta información puede ser desde un simple código o todo un paquete de información guardado en la memoria del TAG.

Para el proyecto se ha escogido el modelo RFID RC522 (Ilustración 10), uno de los más comunes del mercado. Este módulo utiliza un sistema de modulación y demodulación de 13.56MHz, frecuencia que en la actualidad utiliza la tecnología RFID. El módulo se comunica por SCI, de manera que se puede implementar con cualquier microcontrolador con interfaz SPI, como un Arduino, o en mi caso, la placa ESP8266.

Ilustración 10 Lector RFID RC522

2.3.1.5 Raspberry Pi 2

La Raspberry Pi 2 (Ilustración 11) es un ordenador de placa reducida, la cual tiene su propio sistema operativo basado en Debian, denominado Raspbian. Este modelo incluye un procesador modelo BCM2836, con cuatro núcleos de 900MHz. Tiene 1GB de memoria RAM, compartida entre procesador y la tarjeta gráfica. Tienen 4 puertos USB de tipo 2.0, 1 conector RCA y un HDMI. No tiene disco duro interno, sino que se tiene que usar una tarjeta microSD para el almacenamiento. Esta placa dispone de conexión vía Ethernet.

Por las prestaciones descritas anteriormente y las necesidades de este proyecto se ha escogido esta placa porque es la versión más económica que se ha encontrado para la realización del servidor MQTT, conjuntamente con los *scripts* encargados de transformar los datos recibidos a través del servidor para así poder persistirlos en la base de datos.

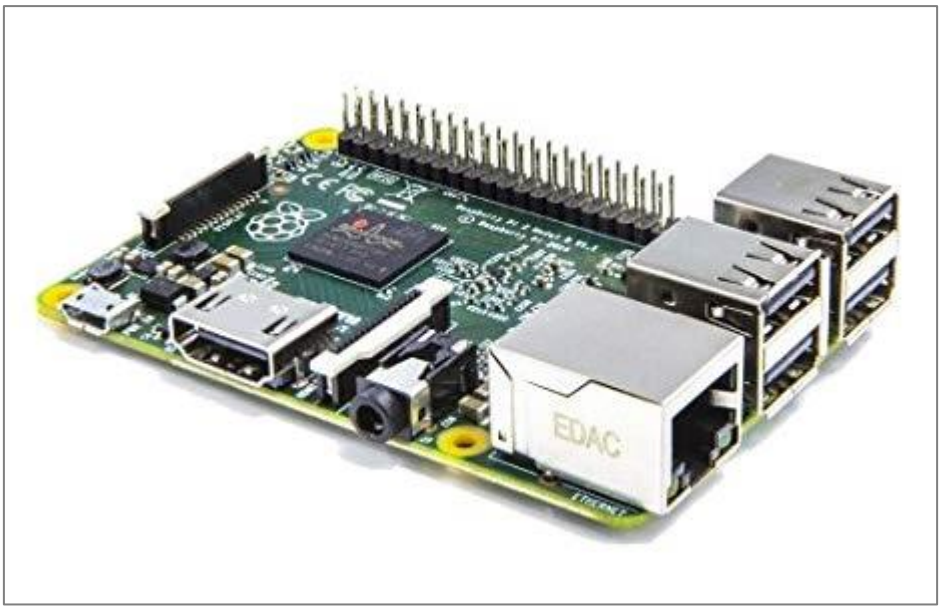

Ilustración 11 Raspberry Pi 2

2.3.2 Tecnologías Software

2.3.2.1 Plataforma software utilizada

Para este proyecto se ha decidido usar Flutter para la implementación de la aplicación, siendo de esta manera una aplicación para plataformas Android e iOS.

Flutter es un framework creado por Google que se utiliza para el desarrollo de aplicaciones Android e iOS. Este usa como lenguaje de programación DART que también se puede usar para

la creación de páginas web. Esto nos permite tener un único código fuente con el que crear aplicaciones para distintos soportes.

DART[14] es un lenguaje nacido el año 2011 de la mano de Google pensado en la optimización para el cliente de aplicaciones rápidas en multiplataforma. Es un lenguaje orientado a objetos que usa un estilo similar al lenguaje C que puede transcompilarse opcionalmente a JavaScript.

2.3.2.2 Herramientas de desarrollo

El IDE [2] que se ha escogido para desarrollar la aplicación del proyecto ha sido **Android Studio** versión 3.4.2 el cual está basado en IntelliJ [2] Aunque este IDE está diseñado para desarrollar aplicaciones nativas en Android, Google ha puesto a disposición de todos un plugin para desarrollar aplicaciones con Flutter.

2.3.2.3 Firebase

Firebase[3] es una plataforma adquirida por Google el año 2014 que permite crear y desarrollar aplicaciones móviles en la nube. Se trata de una plataforma disponible para diferentes plataformas (Android, iOS, web). Tiene una API que permite guardar y sincronizar datos con la nube en tiempo real. También se tiene que remarcar que no es de uso gratuito, ya que tiene 3 niveles de subscripción, aunque en el caso del proyecto se usa el plan gratuito porque no hacen falta más prestaciones.

Firebase tiene dos tipos de bases de datos. Seguidamente se os detallaran algunas de sus características:

- *Realtime Database*
    - Guardar y sincronizar datos entre usuarios en tiempo real.
    - Accesibilidad de los datos des de cualquier dispositivo.
    - Incluida en los SDK para dispositivos móviles, de manera que se pueden crear aplicaciones sin la necesidad de usar servidores.
    - Utiliza la caché local del dispositivo para publicar y guardar cambios que cuando haya conexión sincronizará automáticamente.
    - Integra Firebase Authentication.

- *Cloud Firestone*
    - Guardar, sincronizar y consultar datos a escala global.
    - Se pueden estructurar los datos en colecciones y documentos, incluso creando jerarquías.
    - Se distribuye en SDK para dispositivos móviles y web, de manera que no te hace falta un servidor.
    - Los usuarios pueden realizar modificaciones en los datos sin conexión, y se te notificará de estos.
    - Está diseñada para escalar de manera global.
    - Integra Firebase Authentication.

2.3.2.4 Protocolo MQTT

MQTT**¡Error! No se encuentra el origen de la referencia.** (**M**essage **Q**ueue **T**elemetry **T**ransport) es un protocolo M2M (Máquina a máquina), basado en un protocolo de mensajería publicación/subscripción, al contrario de HTTP que es petición/respuesta.

Uno de sus puntos fuertes es que es extremadamente simple y ligero. Por este motivo es muy interesante para sistemas que requieren poco ancho de banda, tienen una alta latencia y requieren de poco consumo de los dispositivos.

Los objetivos del protocolo MQTT son los siguientes:

- Minimizar el ancho de banda.
- La comunicación bidireccional entre distintos dispositivos.
- Minimizar los requerimientos de los dispositivos, tanto recursos como consumo.
- Garantir la fiabilidad y cierto grado de seguridad.

Los dispositivos utilizan una topología en estrella para conectarse entre ellos, es decir, todos los clientes se conectan directamente con el punto central que hace de servidor. En MQTT este servidor es llamado ***Broker.***

Se trata de una arquitectura basada en eventos. Cada mensaje se envía a los receptores que se hayan subscrito a una publicación concreta. El Broker se encarga de distribuir los mensajes a los receptores. El tópico es el tema donde se subscriben los receptores para recibir el mensaje.

Una de las características más importante es que los clientes o nodos no dependen unos de otros ya que no tienen conocimiento alguno de quien hay al otro lado. Puede incluso que no haya ninguno al otro extremo.

Al contrario de lo que pasa con el protocolo HTTP, no hace falta hacer una petición para recibir información desde un cliente. Cada cliente MQTT abre una conexión permanente TCP con el Broker. Este tiene la capacidad de hacer los mensajes persistentes, guardando los mensajes hasta que el cliente al cual van dirigidos se conecte. El Broker es el único que sabe quién está subscrito a un tópico.

2.3.3 Arquitectura Coldbox

En la Ilustración 12 se puede observar el diagrama que relaciona todas las tecnologías explicadas anteriormente entre ellas.

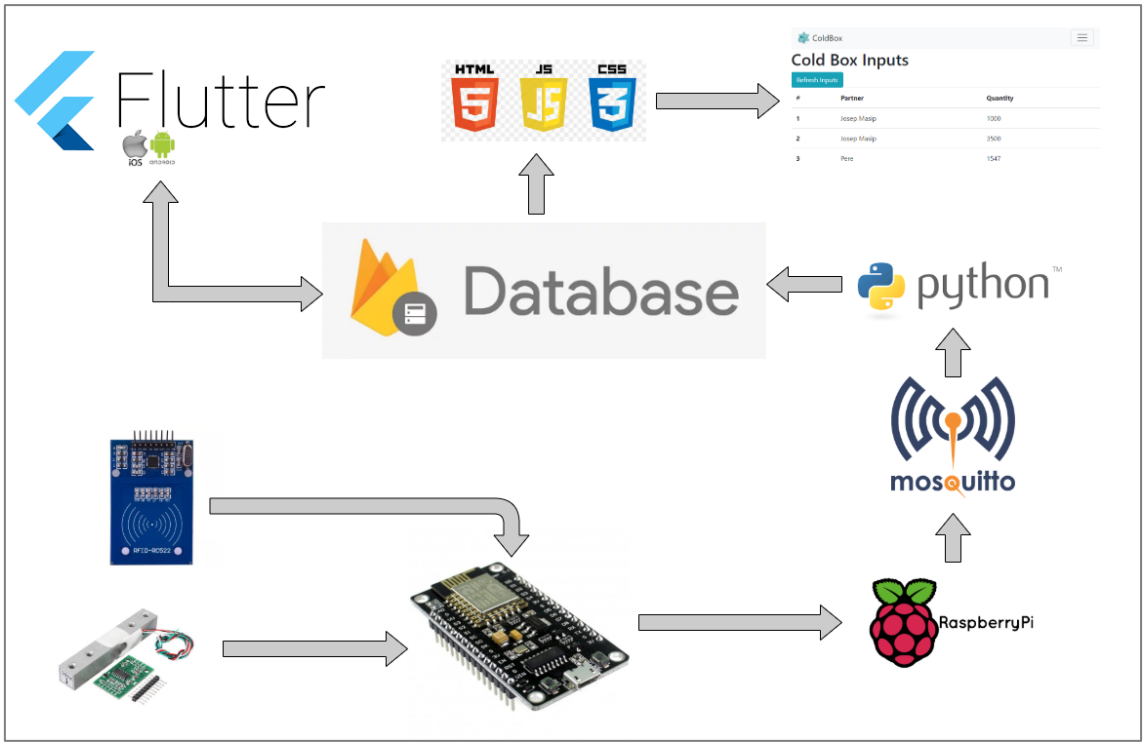

Ilustración 12 Arquitectura Coldbox

Si observamos la imagen de izquierda a derecha, y de abajo hacia arriba, los dos primeros elementos que nos encontramos son el lector RFID RC522 i la celda de carga, los dos están conectados a la placa ESP8266. Esta placa es la encarga de recibir los datos de la celda de carga i el lector y mediante el protocolo MQTT envía los datos al Broker Mosquitto que se encuentra alojado en la Raspberry. Cuando el Broker recibe los datos en script los recoge, comprueba que sean correctos y finalmente envía estos a Firebase, para ser almacenados.

3 Análisis y diseño

3.1 Funcionalidad y análisis

En este episodio se detallan cuáles son los requerimientos funcionales y no funcionales que tendrá el software haciendo un análisis.

Los requerimientos funcionales son los que describen el que el sistema tiene que hacer como respuesta a los eventos que recibe del exterior, incluyendo como tiene que reaccionar a entradas concretas y como se tiene que comportar en situaciones particulares.

En cambio, los requerimientos no funcionales agrupan cualidades generales o propiedades que tiene que exhibir el sistema en realizar su función, reflejan el comportamiento durante la ejecución. En estos requerimientos podemos encontrar atributos de fiabilidad, eficiencia, usabilidad, portabilidad, etc.

3.1.1 Requerimientos funcionales

R.1 Requerimientos funcionales asociados a la creación de un pedido de venta/compra.

- R.1.1 Proporcionar una lista de las diferentes frutas disponibles para crear el pedido de venta/compra.
- R.1.2 Permitir modificar la fecha del pedido de venta/compra.
- R.1.3 Permitir añadir líneas en un pedido de venta/compra.
- R.1.4 Permitir añadir un proveedor/cliente al pedido de venta/compra.
- R.1.5 Permitir añadir quilos al pedido de venta/compra.
- R.1.6 Permitir guardar el pedido de venta/compra a la base de datos permanentemente.
- R.1.7 Permitir cancelar y volver a la pantalla principal.

R.2 Requerimientos funcionales asociados a la ubicación de almacenaje.

- R.2.1 Permitir ver la fruta que hay almacenada.
- R.2.2 Permitir ver la capacidad del almacén.
- R.2.3 Permitir visualizar los quilos de la fruta que hay almacenada.

R.3 Requerimientos funcionales asociados a las gráficas.

- R.3.1 Mostrar una gráfica con la disponibilidad de la fruta que hay.

R.4 Requerimientos funcionales asociados a visualizar otros.

- R.4.1 Mostrar la información de los proveedores/clientes.
- R.4.2 Permitir llamar al proveedor/cliente.

R.5 Requerimientos funcionales asociados a la lista de pedidos de venta/compra.

R.5.1 Permitir visualizar la lista a lo largo de un día deseado.

3.1.2 Requerimientos no funcionales

3.1.2.1 Conectividad

Gestionar los datos de manera descentralizada del dispositivo del usuario, es decir, aunque el usuario cambie de dispositivo, este podrá seguir teniendo todos los datos en otros dispositivos. Esto se consigue gracias a la utilización de los sistemas de almacenado independientes al dispositivo del usuario. En este proyecto, como se ha dicho anteriormente, se usa Firebase.

3.1.2.2 Usabilidad

La aplicación ha de ser fácil de usar y lo más intuitiva posible. Los elementos que formaran parte del menú principal de la aplicación tienen que describir lo mejor posible que puede hacer el usuario en cada parte de la aplicación, y así poder tener una navegación más fluida.

Los pasos para crear un pedido de venta/compra tienen que ser intuitivos y ser lo más claros y directos posibles. Con esto se conseguirá que los pedidos de venta/compra se creen en pocos pasos y el usuario pueda hacer otra acción con rapidez.

Lo que se busca es que el usuario siempre sepa donde esta y no se pierda en ningún momento, para esto se incluye un título en cada pantalla indicando en todo momento en que pantalla se encuentra. También se ha procurado que la interfaz sea lo más gráfica e interactiva posible.

3.1.2.3 Portabilidad

Android es una plataforma que da muchos recursos para hacer que una aplicación sea portable. Da una gran facilidad al programador para poder hacer los diseños más óptimos que se adaptan a diferentes tamaños de pantalla, diferentes idiomas, orientaciones de pantalla, etc.

3.1.2.4 Coste

Todo el software que se usará, Android Studio y Arduino IDE, es totalmente gratuito a efectos personales.

Para poder hacer pruebas físicamente, se usará el dispositivo Android/iOS personal, por lo tanto a nivel de software no habrá gastos.

Pero, como se ha visto anteriormente, sí que los hay a nivel de hardware.

3.1.2.5 Escalabilidad

Android da muchas facilidades para poder escalar una aplicación hacía nuevos dispositivos. Por lo tanto, a la hora de realizar la implementación se dejará preparada para futuras modificaciones que puedan tener la aplicación.

3.1.2.6 Concurrencia

El sistema ha de usar métodos de concurrencia a la hora de realizar las diversas operaciones contra la base de datos. Esto permitirá una mayor fluidez en el hilo principal de la aplicación. El sistema permitirá que diferentes usuarios usen la aplicación.

3.1.2.7 Mantenimiento

El sistema tiene que permitir un fácil mantenimiento de la aplicación. Por lo tanto, se tiene que realizar una buena implementación de todo el código, limpio, entendedor y extensible, para que así sea fácil de mantener en el futuro.

3.1.3 Diagrama de casos de uso

Un caso de uso es una descripción de las acciones de un sistema des del punto de vista del usuario. Es una herramienta valiosa ya que es una técnica de aciertos y errores para obtener los requerimientos del sistema, justamente des del punto de vista del usuario. Los diagramas de casos de uso modelan la funcionalidad del sistema con los actores y los casos de uso. Estos últimos son servicios o funciones provistas por el sistema para los usuarios.

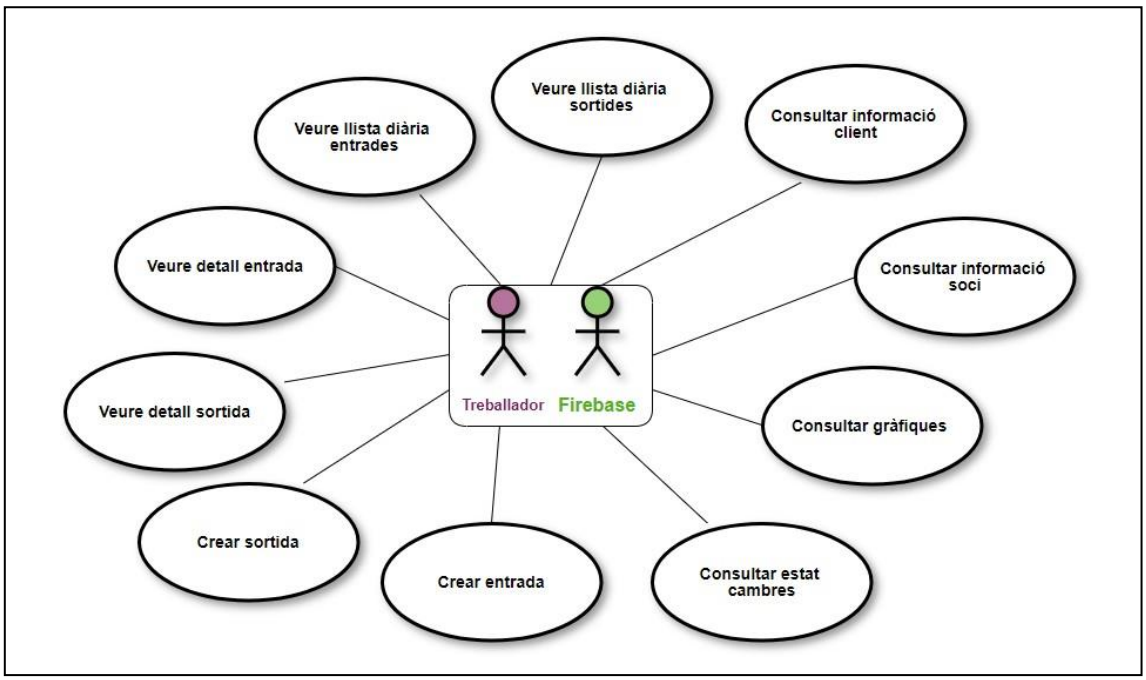

Ilustración 13 Diagrama casos de uso

La Ilustración 13 hace referencia al caso de uso de la aplicación. En ella encontramos el 'Treballador' el cual hace referencia al usuario final de la aplicación, el actor activo, y luego esta 'Firebase' que sería un actor pasivo del caso de uso.

Las elipses representan los distintos casos de uso de la aplicación. Cada uno de ellos describe una funcionalidad del sistema.

Como se puede ver en la Ilustración 13 en todos los casos de uso que hay, participan tanto el actor principal como el pasivo. Ya que el principal es quien siempre inicia el caso de uso, mientras que el pasivo participa en él, en este caso, tratando los datos de cada uno de los casos.

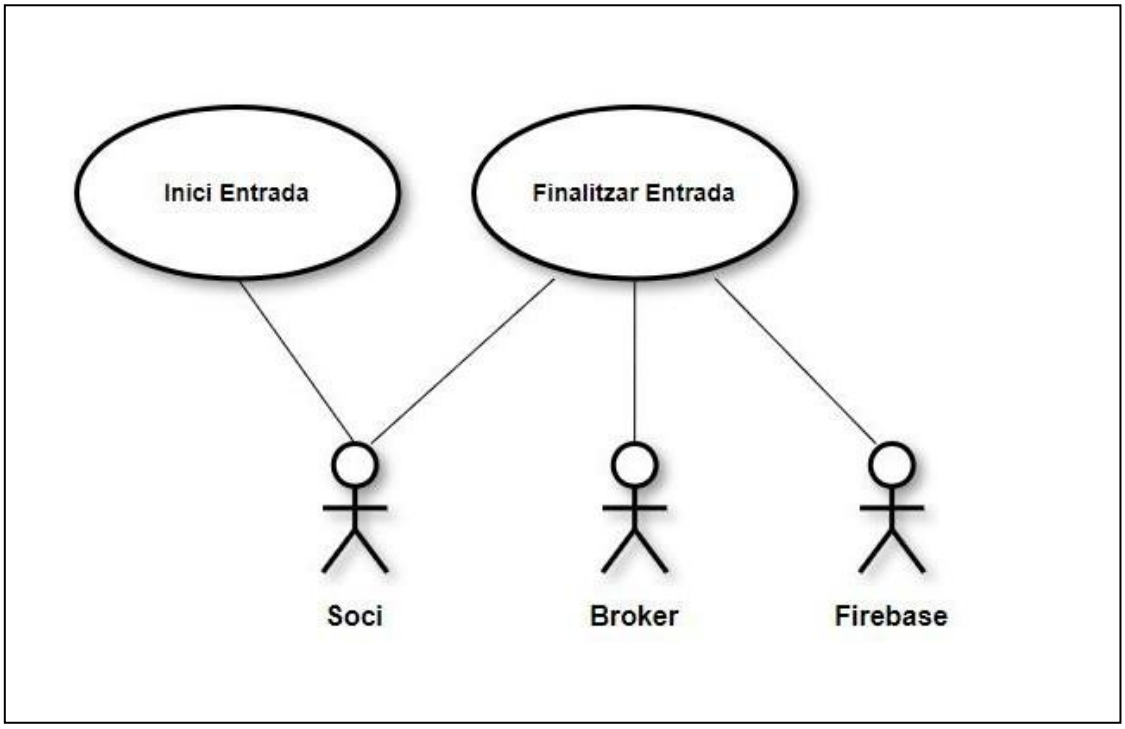

Ilustración 14 Interacción en app

En la Ilustración 14 podemos ver el caso de uso de la báscula. En él participan tres actores. El actor principal es el 'Soci', mientras que el 'Broker' y 'Firebase' son actores pasivos.

Forman parte muy pocos casos de uso en este diagrama, ya que la báscula inteligente tiene una funcionalidad muy reducida.

En ella encontramos los dos casos de uso relacionados con las entradas de fruta. El actor principal participa en ambos, mientras que los actores pasivos solo en el caso de uno "Finalizar Entrada". Se puede ver como en el caso de uso donde participan ambos actores pasivos se hace la comunicación entre la báscula inteligente y la subida de datos a la base de datos. Los datos generados por la báscula inteligente pasan a través del actor 'Broker', mientras que el segundo actor "Firebase" se encarga de la persistencia de datos en la nube.

3.1.4 Especificaciones de los casos de uso

3.1.4.1 Crear entrada

3.1.4.1.1 Especificación de alto nivel

Caso de uso: Crear entrada **Actor/es principal/es:** Usuario (iniciador) i Firebase (ayudante) **Propósito:** Crear una entrada y añadirla automáticamente a la lista. **Descripción:** Un usuario se dispone a crear una nueva entrada. Una vez esta se crea se vuelve

3.1.4.1.2 Especificación expandida

Referencias cruzadas: Requerimientos: R.1.1, R.1.2, R.1.3, R.1.4, R.1.5, R.1.6, R.1.7 **Curso típico de los eventos: (...)**	
Acciones de los actores	**Respuestas del sistema**
1.-El usuario comienza a crear la entra manualmente y pide al sistema los socios y las frutas disponibles para realizar la acción.	
	2.-El sistema muestra los diferentes socios y frutas a elegir.
3.-El usuario selecciona una de las frutas de la cual quiera crear la entrada. El usuario, en caso de añadir otras frutas repite el paso 3 hasta que tenga todas las líneas de la entrada.	
4.-El usuario introduce los quilos entrados.	
	5.-El sistema guarda la entrada a la base de datos.
6. Una vez finalizada la creación de la entrada se vuelve a la pantalla con la lista de ellas.	

Cursos alternativos: 2a: El usuario decide crear la entrada usando NFC 1: Al llegar a la báscula escanea la tarjeta NFC del socio para iniciar la entrada. 2: El sistema da paso a añadir la fruta de esta mediante diferentes tarjetas NFC que pertenecen a cada fruta. 3: El socio vuelve a pasar su tarjeta NFC para así finalizar el proceso de la entrada. **Requerimientos especiales:** No hay ninguno en el curso típico de los eventos, en el curso alternativo es necesario que tanto el dispositivo como el socio, dispongan de una tarjeta NFC.

3.1.4.2 Ver listado diario de entradas

3.1.4.2.1 Especificación de alto nivel

Caso de uso: Ver la lista diaria de entradas. **Actor/es principal/es:** Usuario (iniciador) y Firebase(ayudante) **Propósito:** Mostrar una lista con todas las entradas realizadas en un día seleccionado. **Descripción:** El sistema coge todas las entradas guardadas en 'Firebase' y muestra una lista

3.1.4.2.2 Especificación expandida

Referencias cruzadas: Requerimientos: R.5.1 **Curso típico de los eventos: (...)**	
Acciones de los actores	**Respuestas del sistema**
1.-El usuario se dirige a la lista de entradas. La fecha por defecto es la que tiene el sistema. Para ver la lista de cualquier otro día selecciona una fecha diferente en el calendario que tiene en la pantalla.	2.-El sistema muestra todas las entradas realizadas el día seleccionado.

Cursos alternativos:

1a: El usuario decide ver las entradas de una fecha diferente a la ya seleccionada.

1: El usuario vuelve a escoger otra fecha en el calendario.

2: El sistema muestra todas las entradas con la nueva fecha seleccionada.

Requerimientos especiales: No hay ninguno.

3.1.5 Diagrama de secuencia del sistema

En la Ilustración 15 podemos ver el diagrama de secuencia que nos muestra la interacción entre dos usuarios, socio y trabajador, y el sistema.

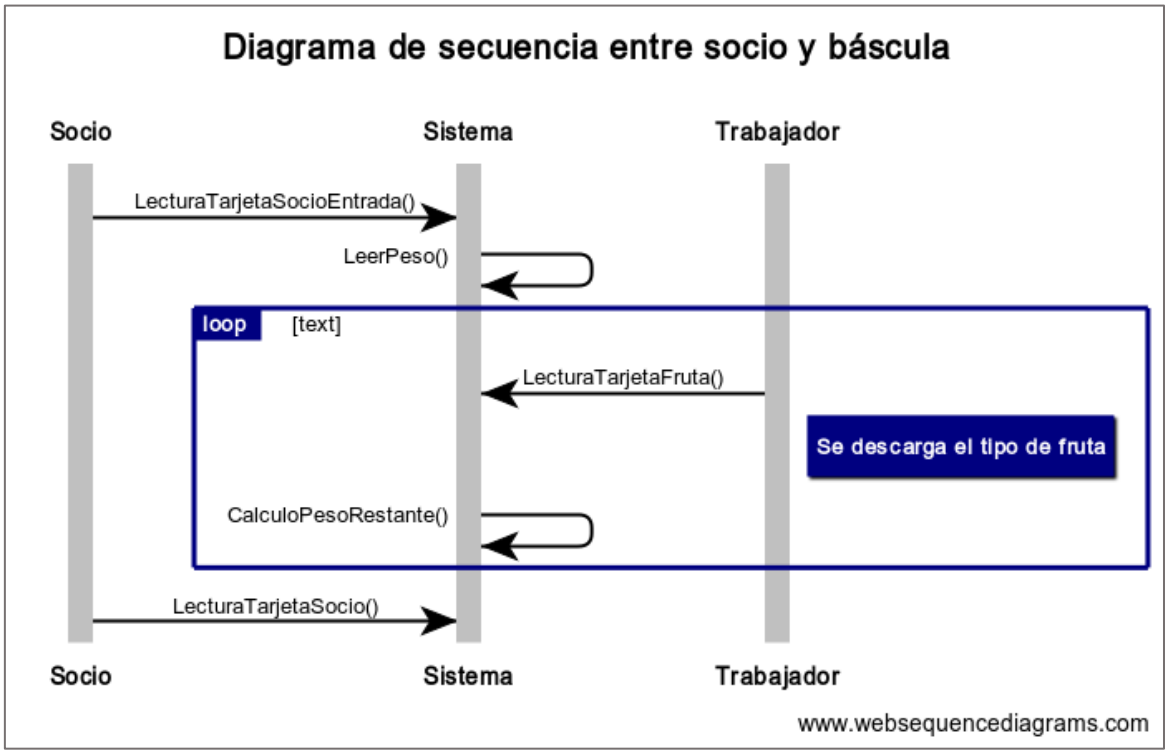

Ilustración 15 DSS Socio y báscula

En esta vemos que el actor que empieza la interacción es el socio, el cual acerca su tarjeta NFC con su identificador único al 'Lector RFID' para que así el sistema pueda leer sus datos, a continuación, el sistema coge el peso total del tractor que viene a descargar el socio.

Seguidamente se entra en el proceso de pesaje, el cual se ejecuta mientras quede fruta a descargar. En este proceso participan el sistema y un trabajador de la empresa, el cual va verificando y asignando un peso a cada tipo de fruta que el socio lleva en el tractor. El

trabajador es el que tiene las tarjetas NFC con la información relacionada con la fruta, y es él, el responsable de que mientras se produce la descarga de fruta ir pasando la tarjeta que corresponda a esta por el 'Lector RFID'. Una vez se haya pesado el primer tipo de fruta que lleva el tractor, se descarga, y en caso de que este lleve más fruta se procede al cálculo del peso restante para volver a reproducir todo el proceso de pesaje. En caso contrario, se pasaría al último paso.

Una vez terminado el proceso, el trabajador da el visto bueno al socio, el cual vuelve a pasar su tarjeta identificativa por el 'Lector RFID', y así termina el proceso.

3.2 Diseño

3.2.1 Estilo de navegación

Basándonos en los dos estilos de menús de navegación que tiene Android se ha decidido usar el menú de pestañas, *bottom navigation* ().

Si comparamos los dos estilos que hay, el de pestañas, y el lateral, *navigation drawer* (), podemos observar que uno es más cómodo y rápido de usar que el otro, ya que lo tienes en todo momento visible en pantalla.

También cabe decir, que también se ha escogido ya que es un menú que últimamente se usa en la mayor parte de aplicaciones del mercado, cosa que facilita al usuario a usarlo ya que este está habituado a él.

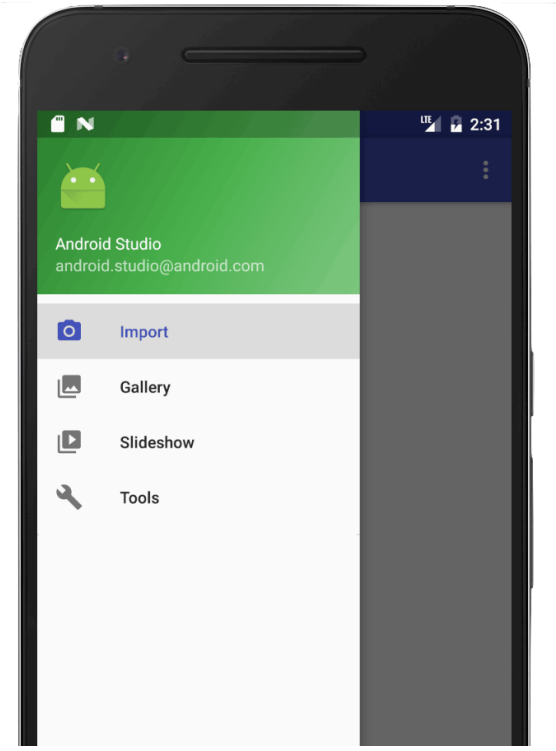

Ilustración 16 Navigation drawer

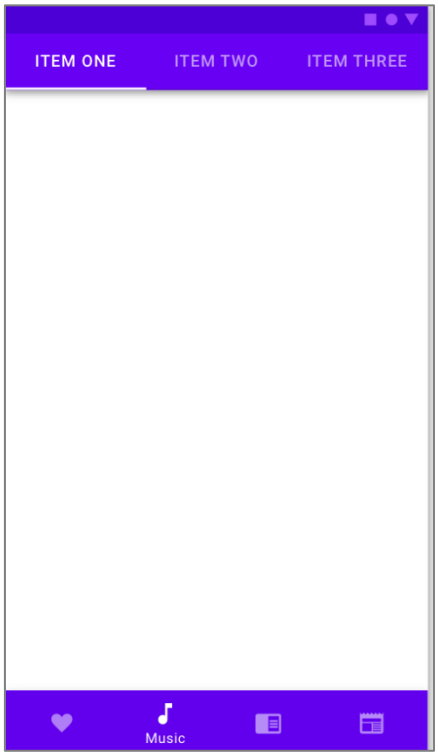

Ilustración 17 Bottom navigation

3.2.2 Mapa de navegación entre pantallas

El diagrama de navegación (Ilustración 18) muestra cómo se organizan las secciones y contenidos de la aplicación. Gracias a ello observamos la estructura de la aplicación.

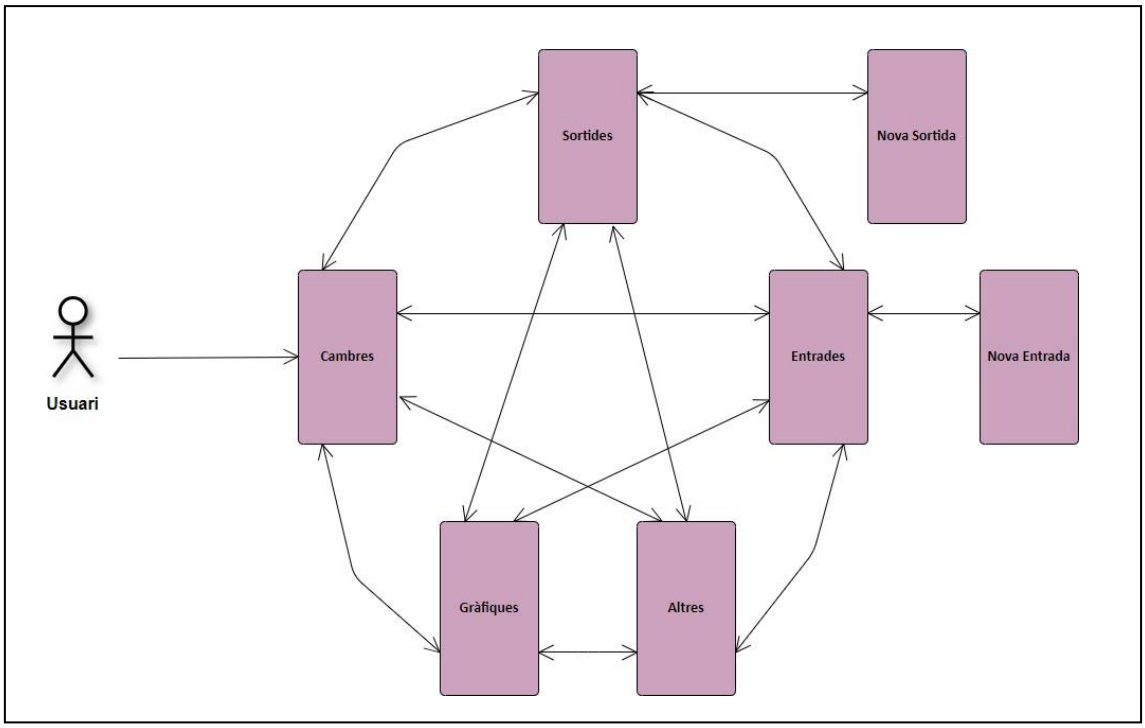

Ilustración 18 Mapa de navegación entre pantallas

En la Ilustración 18 se puede ver el estilo de navegación de la aplicación del proyecto.

Usando como referencia la imagen, podemos ver que en ella hay siete rectángulos verticales rosados, los cuales hacen referencia a cada una de las pantallas que tiene el sistema.

La primera pantalla que se encuentra el usuario al acceder a la aplicación es la de 'Cambres' donde allí puede ver la ocupación de cada cámara y el valor, en kilos, de la fruta o frutas que la ocupan. Des de esa pantalla el usuario puede ir a cualquier otra pantalla de la aplicación.

Las únicas pantallas a las cuales no tiene acceso desde el menú principal son la de 'Crear nueva entrada' y 'Crear nueva salida'. Ya que estas dos son pantallas que dependen de la pantalla principal de 'Entradas', o bien, de 'Salidas'.

3.2.3 Pantallas

3.2.3.1 Listado de entradas

En esta pantalla se muestra una lista, ordenada por número de entrada, juntamente con el nombre del socio que ha hecho la descarga de fruta y los kilos de fruta que ha entrado. También en la parte superior de la pantalla hay un calendario, donde por defecto tenemos seleccionada la fecha del día en el que estamos. Este día puede ir cambiando y así poder ver las entradas de los diferentes días. En la barra superior, al lado derecho, existe un botón el cual te redirige al formulario para crear una nueva entrada.

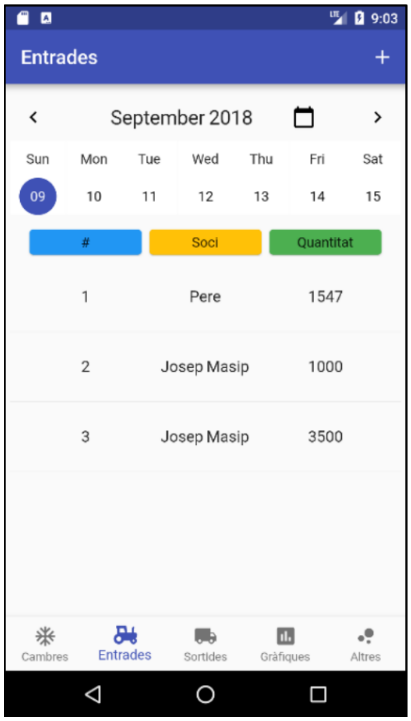

Ilustración 19 Listado de entradas

3.2.3.2 Listado de salidas

En esta pantalla se muestra una lista, ordenada por número de salida, juntamente con el nombre del cliente que ha hecho la compra y los kilos de fruta que ha comprado. También en la parte superior de la pantalla hay un calendario, donde por defecto tenemos seleccionada la fecha del día en el que estamos. Este día se puede ir cambiando y así poder ver las salidas de los diferentes días. En la barra superior, al lado derecho, existe un botón el cual te redirige al formulario para crear una nueva salida.

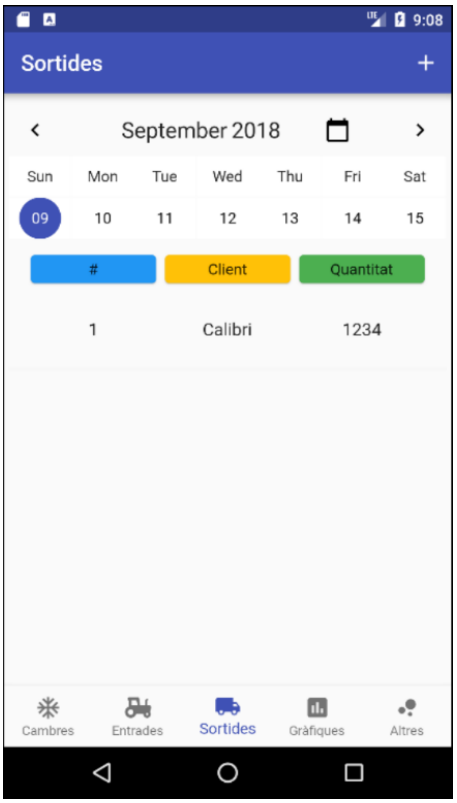

Ilustración 20 Listado de salidas

3.2.3.3 Crear nueva entrada

En esta pantalla es donde el usuario des de la aplicación puede crear una nueva entrada. Esta se podrá hacer manualmente o bien mediante una tarjeta NFC que tendrá el socio.

- **Manualmente**

  En el caso de hacer la entrada manualmente, el usuario tendrá que introducir el nombre del socio, la fruta y los kilos descargados y guardar la operación. La fecha de la entrada es automática, aunque se da la opción de poder cambiarla.

- **NFC**

  Este proceso es el que pertenece a la Ilustración 15.

3.2.3.4 *Cámaras*

En esta pantalla sale un listado de todas las cámaras frigoríficas de las que dispone la cooperativa. En ellas se organizan los kilos de fruta que van entrando y así el usuario puede ver con facilidad donde está ubicada cada fruta. En el caso de que en alguna cámara no disponga de fruta esto se indica con un mensaje para el usuario.

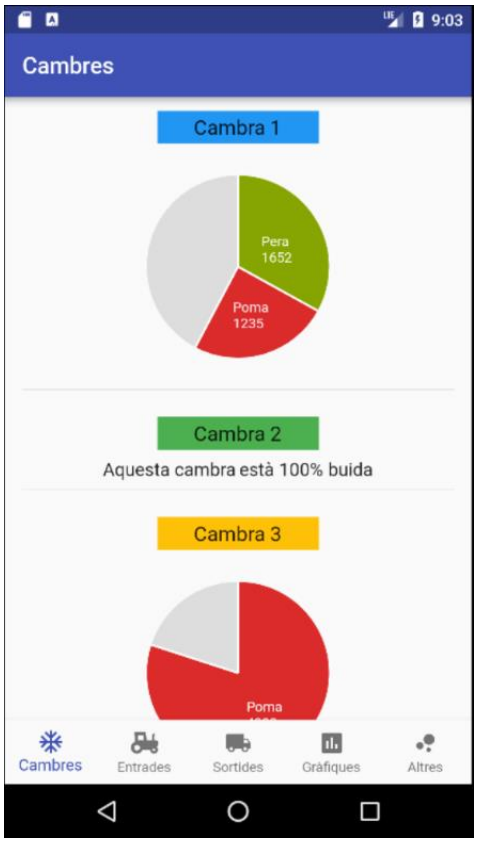

Ilustración 21 Cámaras

3.2.3.5 *Gráficas*

En esta pantalla aparecen gráficas relacionadas con los movimientos de fruta que ha habido durante un periodo de tiempo. Cada una de ellas tiene un filtro diferente, ya sea diario, semanal o anual.

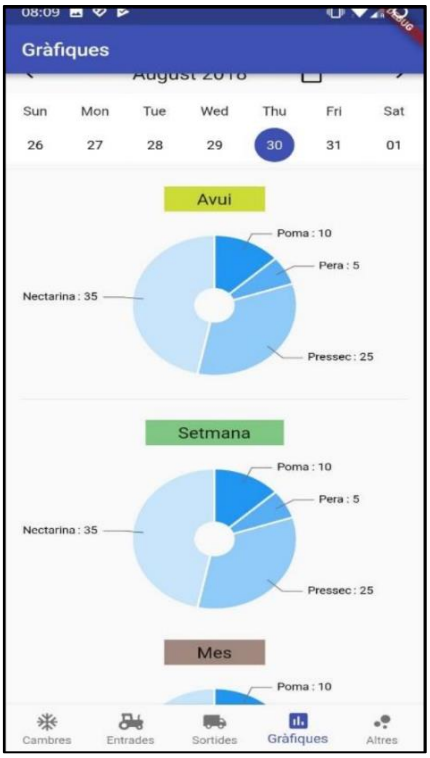

Ilustración 22 Gráficas

3.2.3.6 *Otros*

Esta pantalla está dividida en dos pestañas. Así el usuario puede navegar desde la pestaña con los datos de los socios a la pantalla con los datos de los clientes. En estas podemos ver que tenemos los datos fiscales de los socios y de los clientes, los cuales no se pueden modificar ya que vienen dados y la única acción que podemos realizar es la de la llamada.

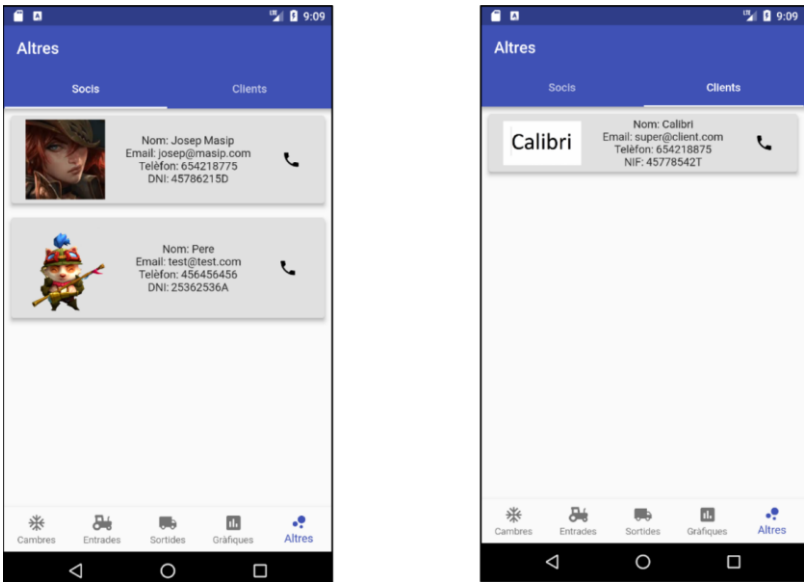

Ilustración 23 Pantalla datos socios y clientes

4 Decisiones e implementación

Después de haber explicado las tecnologías a utilizar, haber hecho un análisis de requerimiento y explicado el diseño de todas las pantallas, ya podemos empezar a explicar las decisiones tomadas en el proyecto, así como la implementación de la aplicación Coldbox y la implementación de la báscula inteligente.

4.1 Decisiones tomadas

4.1.1 Versión API Android

Primeramente, tuvimos que plantearnos una característica muy importante en una aplicación. Es muy importante decidir que versión de API se va a utilizar, ya que esto influye en la compatibilidad de la aplicación con los dispositivos móviles. A parte de esto también entra en juego el parámetro *compileSdkVersion.* Este parámetro indica al compilador de Android Studio en que versión tiene que compilar el código y por lo tanto indica que funcionalidades del sistema se podrán usar en el código. El valor de este parámetro tendrá que ser siempre el mayor posible para poder dotar la aplicación con todos los elementos posibles. Por ejemplo, si se establece el valor de *compileSdkVersion* en 16 se puede ejecutar la aplicación en un dispositivo con API 15, siempre y cuando no se intente invocar métodos específicos de la API 16.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.2%
4.2.x		17	1.5%
4.3		18	0.5%
4.4	KitKat	19	6.9%
5.0	Lollipop	21	3.0%
5.1		22	11.5%
6.0	Marshmallow	23	16.9%
7.0	Nougat	24	11.4%
7.1		25	7.8%
8.0	Oreo	26	12.9%
8.1		27	15.4%
9	Pie	28	10.4%

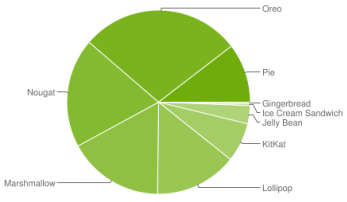

Ilustración 24 Camparativa uso de APIs Android

Basándonos en el gráfico anterior (Ilustración 24) se ha escogido la Api 16, que hace referencia a Android 4.1 *Jelly Bean,* para poder llegar al máximo nivel de usuarios. Esto también se ha hecho pensando en la necesidad del usuario de poder reaprovechar un dispositivo viejo sin la necesidad de conseguir uno nuevo.

4.1.2 Versión iOS

Al ser una aplicación multiplataforma hemos de tener en cuenta las versiones de los dos sistemas operativos en los que va a funcionar. Anteriormente hemos comentado que versión de API Android se ha escogido y el motivo de la elección.

Para iOS se ha escogido la versión 8, ya que como se puede ver en la Ilustración 25, esta versión está comprendida en el 5% que representa las versiones 'Anteriores', por lo tanto, nuestra aplicación vendría a ser compatible con todos los dispositivos iOS, con versión 8 o superior.

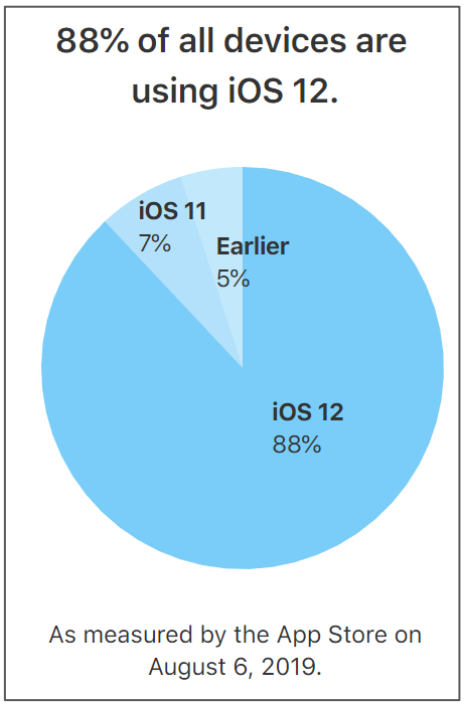

Ilustración 25 Gráfico uso versiones iOS

También se ha tenido que tener en cuenta la versión iOS a la hora de implementar, ya que todos los elementos usados en el diseño de la interfaz tienen que ser compatibles con esta versión. Además de los elementos de diseño, también se tienen que tener en cuenta las funcionalidades que conlleva cada versión.

4.2 Estructura de la base de datos

Como base de datos estamos usando Firebase Firestore que como bien hemos comentado es una base de datos NoSQL. Este tipo de base de datos no funciona con relaciones, sino que usa Colecciones y Documentos. Para el proyecto Coldbox se han usado las siguientes Colecciones para almacenar los datos generados tanto por la báscula como por la aplicación móvil.

- Colección Fruits

En la colección Fruits se están guardando todos los datos de las diferentes frutas. Cada fruta está representada a partir de un documento con la *id* aleatoria. Los atributos elegidos para ser guardados han sido el color, su color en hexadecimal, el nombre, el precio, el color de la piel y su variedad.

- Colección Rooms

En la colección Rooms se están guardando todos los datos de las diferentes cámaras de fruta que hay en la cooperativa. En la Ilustración 26 podemos ver como cada cámara está representada a partir de un documento con la *id* aleatoria. Los atributos elegidos para ser guardados han sido la capacidad total de la cámara y una lista del género almacenado. El género está formado por dos atributos, una referencia a la fruta almacena y cuantos kilos hay de esta.

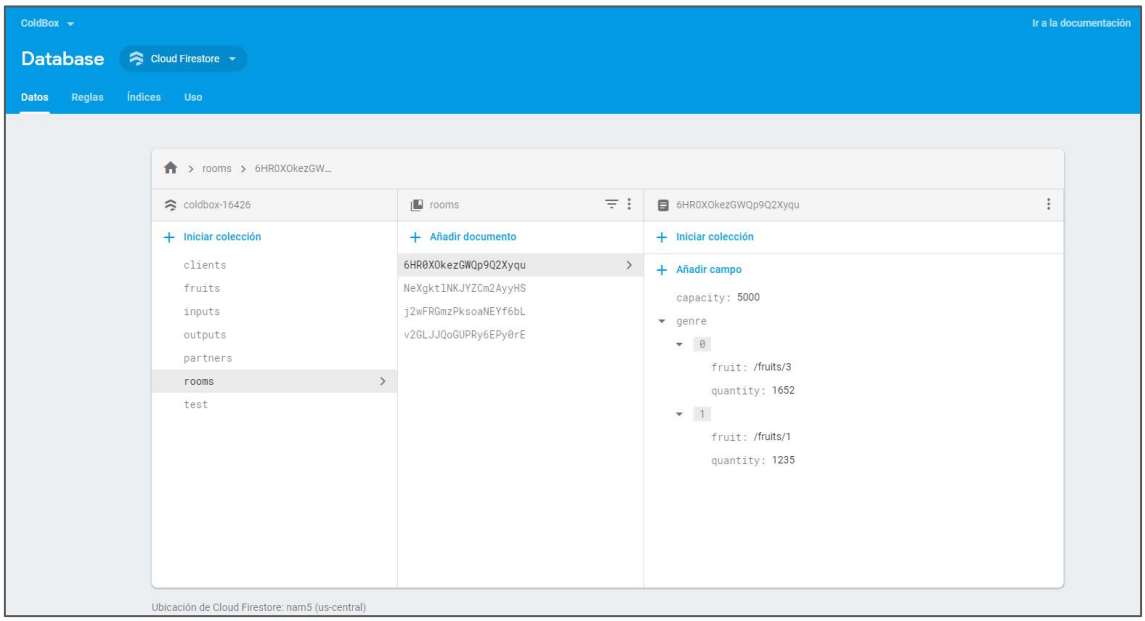

Ilustración 26 Estructura colección Rooms

- Colección Partners

En la colección Partners se están guardando todos los datos de diferentes socios que tiene la cooperativa. Cada socio está representado a partir de un documento con una *id* aleatoria. Los atributos elegidos para ser guardados han sido el correo electrónico, el CIF del socio, su nombre y su número de teléfono.

- Colección Clients

En la colección Clients se están guardando todos los datos de los diferentes clientes que hay en la cooperativa. Cada cliente está representado a partir de un documento con un *id* aleatorio. Los atributos elegidos para ser guardados han sido muy similares a los atributos de los socios, el email, el CIF del cliente, el nombre, el número de teléfono y su imagen corporativa.

- Colección Inputs

En la colección Inputs se están guardando todos los datos de las entradas que se hacen en la cooperativa. En este caso, se están creando varios documentos donde sus id corresponde al timestamp en milisegundos del día de la entrada. Estos documentos a su vez contienen una colección también llamada Inputs. En esta última colección se almacenas los documentos de las diferentes entradas con un id aleatorio. De esta manera tenemos todas las entradas ordenadas por el día que se han realizado. Los atributos elegidos para ser guardados en los documentos de las entradas han sido la fecha y hora de cuando se ha realizado la entrada, una referencia al socio y una lista del género entrado. Esta lista tiene el mismo formato que en las cámaras, cada elemento tiene dos atributos, una referencia a la fruta entrada y la cantidad de esta.

- Colección Outputs

En la colección Outputs se están guardando todos los datos de las salidas que se hacen en la cooperativa. En este caso, se están creando varios documentos donde sus id corresponde al timestamp en milisegundos del día de la salida. Estos documentos a su vez contienen una colección también llamada Outputs. En esta última colección se almacenas los documentos de las diferentes salidas con un id aleatorio. De esta manera tenemos todas las salidas ordenadas por el día que se han realizado. Los atributos elegidos para ser guardados en los documentos de las salidas han sido la fecha y hora de cuando se ha realizado la salida, una referencia al

cliente y una lista del género salido. Esta lista tiene el mismo formato que en las cámaras, cada elemento tiene dos atributos, una referencia a la fruta salida y la cantidad de esta.

4.3 Implementación de la aplicación

4.3.1 MVC

En nuestra implementación en Flutter utilizamos el patrón de arquitectura llamado Modelo Vista Controlador (MVC). La principal característica del MVC consiste en separar los datos de una aplicación, su interfaz de usuario y la lógica en tres componentes diferentes que se relacionarán entre si y así conseguir un código modular fácil de mantener. Flutter no tiene una buena estructura definida para implementar este patrón como la puede tener una aplicación Android nativa, aun así, se ha implementado en la mayor medida posible.

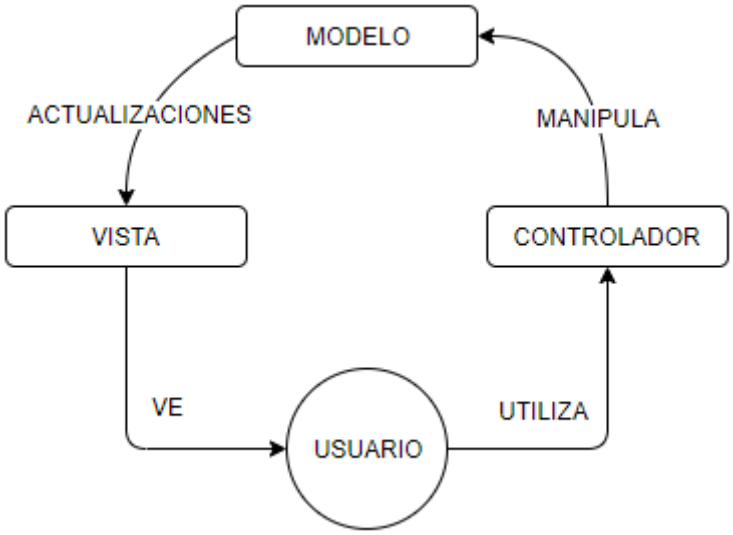

Ilustración 27 Modelo-Vista-Controlador

- **M**odelo: Representa el modelo de datos de nuestra solución, el cual dispone de la información del mundo real que el sistema debe reflejar. Es la parte encargada de representar la lógica de negocio de una aplicación.
- **V**ista: Las vistas son las encargadas de la representación de los datos, contenidos en el modelo, al usuario. Contienen todo lo necesario para mostrar de forma visual los diferentes modelos de datos que contenga la aplicación.
- **C**ontrolador: Es el elemento que permite realizar la interacción entre las acciones del usuario sobre nuestra aplicación, pueden manipular el modelo de datos y actualizan la vista según las acciones que se lleven a cabo.

De esta manera podemos dedicarnos a desarrollar nuestros componentes de tal manera que incluso podríamos reutilizarlos en proyectos posteriores y no simplemente en el proyecto actual. Para lograr esto, el diseño de la implementación de nuestra aplicación juega un papel importante y la capacidad de abstracción que tengamos desarrollada. En la aplicación podemos ver este patrón reflejado a la perfección.

Los modelos de la aplicación se representan en las clases *ClientPartner*, *Fruit*, *FruitQuantity*, *Input* y por último *Room*. Estas clases son las responsables de representar los diferentes modelos en la aplicación de Coldbox.

A continuación, se van a explicar las diferentes clases modelo, y así poder comprender mejor el modelo de datos que representa la información manejada por la aplicación.

La clase *ClientPartner* se encarga de definir el modelo de datos de las colecciones *Clients* y *Partners* que se va a usar para tanto el socio como el cliente, ya que los dos usan los mismos campos.

La clase *Fruit* se encarga de definir el modelo de datos de la colección Fruits.

La clase *FruitQuantity* se encarga de definir una pareja clave, valor donde se especifica la fruta como la clave de la pareja y los kilos como valor para cada entrada y/o salida.

La clase *InputOutput* se encarga de definir el modelo de datos de las colecciones *Inputs* y *Outputs,* que usan los mismos campos.

La clase *Room* se encarga de definir el modelo de datos que se usa para la colección *Room.*

4.4 Implementación de la báscula

Una vez explicada la implementación de la aplicación móvil, vamos a explicar la implementación de la báscula inteligente. Gracias a esta podemos automatizar todo el sistema de pesaje, desde que el tractor llega a la cooperativa hasta que se finaliza la descarga.

La báscula inteligente consta de dos partes de igual importancia. La primera parte hace referencia a la báscula en sí, el código necesario para que funcione todo el sistema. La segunda parte hace referencia al programa encargado de tratar todos los datos generados por una o varias básculas. En este proyecto solamente se ha usado una báscula, ya que no había

presupuesto para una segunda, aunque el sistema funcionará perfectamente si hubiera existencia de una segunda o más.

4.4.1 Báscula

Para la báscula se ha usado una placa ESP8266 como cerebro de esta. Juntamente a la placa, se ha puesto una celda de carga la cual se va a encargar de hacer el pesaje.

Se podría usar un Arduino como cerebro de la báscula, ya que ambos usan un chip muy parecido y pueden hacer prácticamente lo mismo. La gran diferencia entre usar un Arduino y usar la ESP es la conectividad WIFI. Con ella vamos a poder enviar los datos a través de MQTT. La placa ESP8266, se ha programado directamente con el IDE de Arduino, usando el lenguaje C para la programación.

4.4.2 Script de gestión

La segunda parte importante del sistema es el script de gestión de datos provenientes de la/s báscula/s. En este reside toda la responsabilidad de procesar los datos y enviarlos posteriormente a la base de datos, en nuestro caso a Firebase Firestore.

El script ha sido implementado en Python 3.7 dada su sencillez y las librerías que nos proporciona Google para hacer la integración con la base de datos.

```
1  {
2    "partner":"HJKI68LK9",
3    "genre":{
4      "fruit_1":2987,
5      "fruit_2":4938
6    }
7
8  }
```

Ilustración 28 Json en formato String

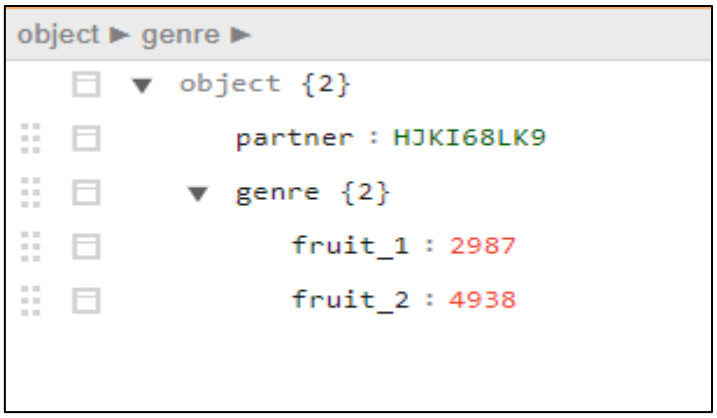

Ilustración 29 Json en visor online

En las ilustraciones anteriores, Ilustración 28 e Ilustración 29, podemos ver la estructura Json que se ha usado para el mapeo de datos. Se ha buscado una estructura sencilla pero a la vez capaz de representar todos los datos necesarios para las entradas y salidas.

En el Json podemos ver los *partners* y *genre* que representan el socio que hace la entrada y la fruta de la cual hace la entrada juntamente con la cantidad de kilos introducidos.

5 Conclusiones

5.1 Conclusión

Ha estado una experiencia muy buena el poder realizar este trabajo, aunque en muchos momentos me he sentido con demasiada presión y trabajo. Pienso que se ha realizado un buen trabajo, aunque no tanto como deseaba en un inicio. Me ha costado más de lo que pensaba y tampoco tenía pensado hacer lo que he terminado haciendo, y esto me ha quitado mucho tiempo. He disfrutado mucho haciéndolo e incluso me ha gustado el poder hacer una parte práctica, ya sea el desarrollo de la aplicación como el montaje de la báscula inteligente, aunque este último ha sido todo un reto para mí. Creo que, si hubiera dedicado más tiempo, habría quedado muy diferente al resultado final.

Se ha incorporado Firebase a la base de datos, y esto da un valor añadido ya que es una base de datos nueva y desconocida por mi parte. Pienso que un proyecto como este si se mejorara podría tener futuro en el mundo frutícola, el cual es un mundo muy grande en la zona en la que vivimos y alrededores.

En los inicios el realizar la parte física del proyecto no era una de las prioridades, pero a medida que ha pasado el tiempo he pensado que estaría bien poder tener un prototipo de la báscula inteligente para apoyar el funcionamiento de la aplicación.

5.2 Trabajo futuro

Después de realizar todo el proyecto, aun se podrían añadir más cosas para complementarlo. Son mejoras que mejorarían mucho el proyecto y que incluso harían que se pudiera valorar la entrada del proyecto en el mercado. En estos momentos todo lo que se tiene es un prototipo simple. Con las siguientes mejoras podría ser un prototipo mucha más efectivo.

- Traducciones para la aplicación

  En estos momentos la aplicación esta únicamente en catalán. Se tendría que añadir traducciones para otros idiomas, como mínimo el castellano y el inglés.

- Mejoras de seguridad

  La seguridad del proyecto en estos momentos es nula. Se tendría que poner más seguridad entre la báscula inteligente y la aplicación, cifrar las conexiones, poner certificador de

autentificación en los dispositivos que se conecten y pedir una autenticación entre el servidor Broker y sus clientes.

Implementar un inicio de sesión para poder verificar que únicamente usuarios permitidos pueden acceder y usar la aplicación.

Añadir reglas a la base de datos para restringir el uso solamente a personal autorizado.

- Mejoras en la implementación de la aplicación

Hay algunas pantallas que su diseño final no ha sido como el que se deseaba al inicio. Por ejemplo, la pantalla de cámaras (3.2.3.4) ya que la intención inicial de la pantalla era crear un mapa interactivo el cual no se ha podido conseguir por manca de experiencia y de conocimientos de programación.

- Implementación aplicación web

En el inicio del proyecto se pensó en implementar una aplicación web complementaria a la aplicación móvil, ya que en el móvil solo se pueden crear y visualizar datos. La aplicación web habría permitido la modificación de datos y también la eliminación, además, esta permitiría añadir datos adicionales que no estén en la aplicación móvil, como, por ejemplo datos de facturación de clientes y socios o también, datos de nuevos clientes y socios.

6 Tabla de referencias

[1] IDE - https://es.wikipedia.org/wiki/Entorno_de_desarrollo_integrado

[2] IntelliJ - https://www.jetbrains.com/idea/

[3] Firebase – https://firebase.google.com/

[4] Cooperativa del Camp Sant Gaietà - http://www.fruitona.com/

[5] Google Play - https://play.google.com/store

[6] NFC - http://nearfieldcommunication.org/

[7] Flutter - https://flutter.dev/

[8] NodeMCU - https://www.nodemcu.com/index_en.html

[9] Arduino - https://www.arduino.cc/

[10] Wi-Fi Alliance - https://www.wi-fi.org/

[11] Puente Wheatstone - https://www.ecured.cu/Puente_de_Wheatstone

[12] Conversor A/D - http://www.ifent.org/Lecciones/digitales/secuenciales/ConvertA_D.htm

[13] RFID RC522 - https://naylampmechatronics.com/blog/22_Tutorial-Lector-RFID-RC522.html

[14] Dart - https://dart.dev/

[15] MQTT - http://mqtt.org/

[16] Gantt Project - https://www.ganttproject.biz/

[17] Versión API Android e IOS - https://flutter.dev/docs/resources/faq#what-devices-and-os-versions-does-flutter-run-on

[18] Wireframe - https://www.experienceux.co.uk/faqs/what-is-wireframing/